C++ Programming Style (C++编码规范)

Last Modified: 2007-10-05
Hongfei Yan

=目的**=**
增强代码的可读性, 从而使得系统具有更少的Bug和更高的健壮性。涉及到程序的书写格式(Code appearance)、标示符命名(Naming)、
、文档规范(注释)、模块组织规范和一般性原则。

=书写格式规范**=**

1. place the brace at the end of the line that controls entry into the block
(左花括号写在行尾代码块的开始)
For-loop statements:

```
for (int i = 0; i <= j; i++) {
  // ...
}
```

If and else statements:

```
if (j < 0) {
  // ...
}
else {

  // ...

}
```

2.Use a single space to separate the keywords, parentheses, and curly braces in conditional statements.
(条件语句中, 比如if() for() while() try(), 使用一个空格分隔关键词, 括号和花括号。)

```
for.( ...).{
  // ...
}
```

3. 不使用"hard"TAB
配置好编辑器, 把对齐缩进时敲入的hardTab保存为空格。缩进使用4个空格。
VIM

```
:set tabstop=4
:set expandtab
```

=标示符命名规范**=**

1. 使用有意义的, 一致的名字, 不使用缩写

```
if (age < RetirementAge) {
  yearsToRetirement = RetirementAge - age;
}
else {
  yearsToRetirement = 0;
}
```

2. 类, 常数, 结构, 枚举和类型名使用“UpperCamelCase”==

```
enum BackgroundColor {
  None,
  Red,
  Green,
  Blue
};
const int FixedWidth = 10;
class BankAccount {
  // ...
};
typedef list<BankAccount> Portfolio;
```

3. 函数名, 变量名, 参数名, 缩略语使用“lowerCamelCase”

```
class Customer {
  public:
    void setAddress(const Address& address);

};
int fixedWidth = 10; // it’s a variable, not a constant
XMLString -> XmlString
loadXMLDocument() -> loadXmlDocument()
```

4. 定义对象的类、结构和类型名使用名词

```
class Customer {
  // ...
};
typedef int Dollars;
```

5. 变量名使用名词

```
class Customer {
  private:
    Address m_billingAddress;
    Address m_shippingAddress;
    Phone m_daytimePhone;
    Orders m_openOrders;
};
```

6. 函数名使用动词

```
class Account {
  public:
    void withdraw(int amount);
    void deposit(int amount);
};
```

7. 不要省略参数名

```
MyClass(int);                // Bad
MyClass(float meaningfulName);  // Good
```

=文档规范=

1. 文件头的版权和描述信息:每一个头文件(.hpp)包含一个如下所示的文件头:

```
/**
 * @file filename
 * @brief Brief description goes here.
 * @author Gailun Zhang<zgl@gmail.com>
 * @version 0.1
 * @history
 *    1. created on 2007-10-5
 *    2. write your history2 here
 */
```

2. 使用block注释描述代码接口(program interface):类定义中每一个成员函数前都包含如下注释:

```
/**
 * @brief Brief description goes here.
 * @param argOne Description of first function argument.
 * @param argTwo Description of second function argument.
 * @return Description of returned value.
 */
```

3. 不在行尾写注释

```
double length = 0;    // vector length.       BAD
```

=模块组织规范=

分离接口与实现是C++模块管理的基本思想。概念上的分离应该保证:用户只看到他所关心的接口,而看不到实现。

1. 使用类名作文件名

一般一个文件只包含一个public的类。文件名使用类名,并使用同样的大小写风格(UpperCamelCase)。对应的头文件扩展名为.h,

实现文件扩展名为.cpp。

2. 头文件中不使用全局作用域的using或using namespace语句

在非全局作用域内,可自由使用using语句,比如自己定义的namespace中,和.cpp的类实现文件中,它们不会影响其它的代码。

3. 在头文件内部而不是在外部写包含检查

建议使用_ModulePrefix_filename_INCLUDED_这样的命名作为include guard的名称。

```
// in foo.h
#ifndef APP_FOO_H_INCLUDED_
#define APP_FOO_H_INCLUDED_
// … contents of the
#endif
```

=一般性原则=

1. 初始化所有变量

```
// Acceptable: Create an empty path
char path[MAX_PATH]; path[0] = '\0';
// Better: Create a zero-filled path
char path[MAX_PATH] = {'\0' };
```

2.一行不要太长,不能超过显示区域.太长则应折行,折行最好发生在运算符前面.

3."Hungarian" prefixes:
m_ Variable is a member of class.
a array
c count / number of
cb count bytes / number of bytes
d Data type double
dw Data type double word
e element of an array
b boolean
g_ global variable
s_ Static variable
l long data type. i.e. "long int"
n int
u unsigned
str C++ GNU template data type "string"
p pointer
lp pointer to long data type
ch C character
sz Zero terminated string. char[]
min Lowest value
first First element
cmd Command
CAbcXyz abcXyz

4. 尽量以const和inline取代#define

5. 尽量以<iostream>取代<stdio.h>

参考文献
[Yang, et al., 2007] TFS-C++ Coding Standards,http://docs.google.com/Doc?id=dg7ww2d7_47fd6gjr&invite=dn5kcb2
[encyclopedia, 2007] Programming style, http://encyclopedia.thefreedictionary.com/coding+style
[yolinux, 2007] http://www.yolinux.com/TUTORIALS/LinuxTutorialC++CodingStyle.html
[Meyers, 1998] Scott Meyers, *Effective C++: 50 Specific Ways to Improve Your Programs and Design (2nd Edition)*: Addison-Wesley, 1998.